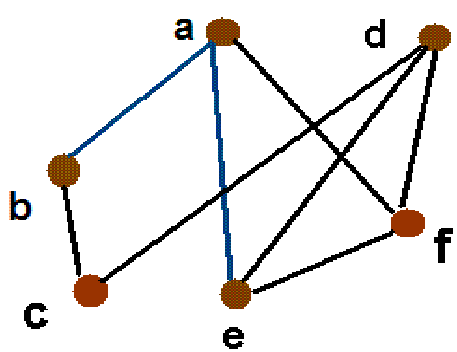
ARBOLES Y GRAFOS

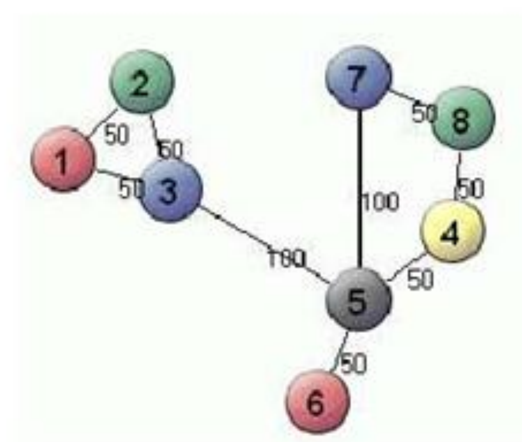
Rosa Barrera Capot
rosa.barrera@usach.cl

¿Grafo?

Es un conjunto de puntos -nodos o vértices- unidos por líneas –arcos o aristas-.



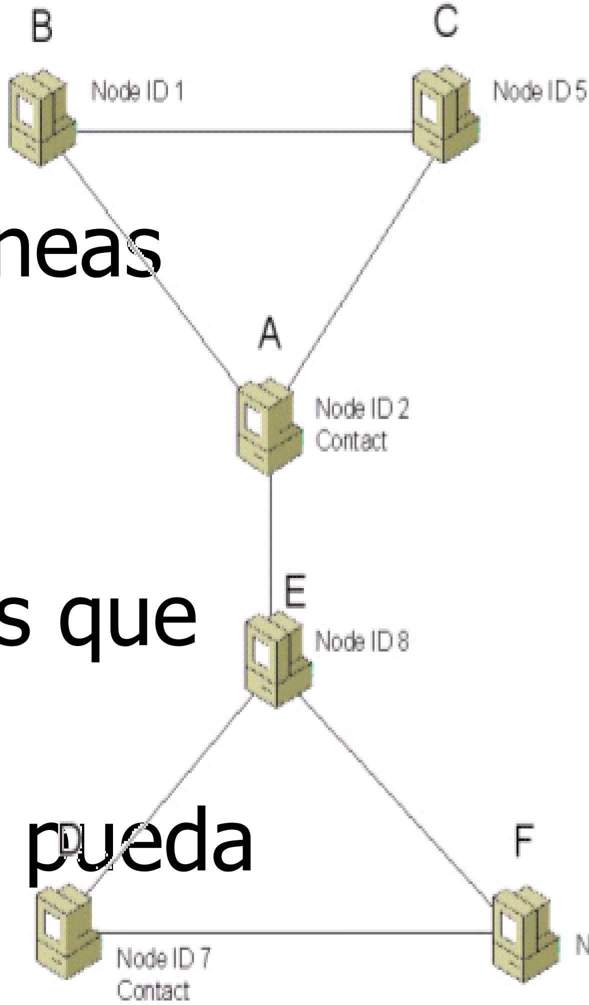
G



Características

- Permiten Modelar un problema
- Aplicaciones:
  - Ingeniería de Sistemas
  - Modelado de Redes
  - Ingeniería Industrial
  - Química
  - Geografía
  - etc

Representaciones más usadas



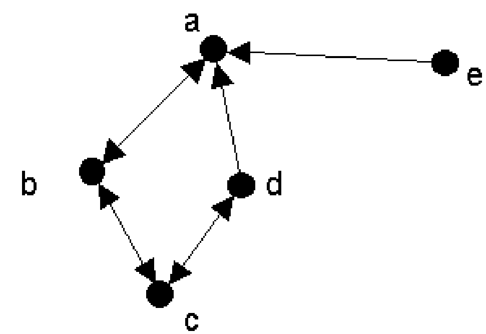
- Red de computadores
- Conexiones de vuelo de aerolíneas
- Carreteras que unen ciudades
- Circuitos eléctricos
- En más pequeño, las máquinas que reciben pago por “algo”
- Cualquier problema que se les pueda ocurrir!!!

Definición Formal

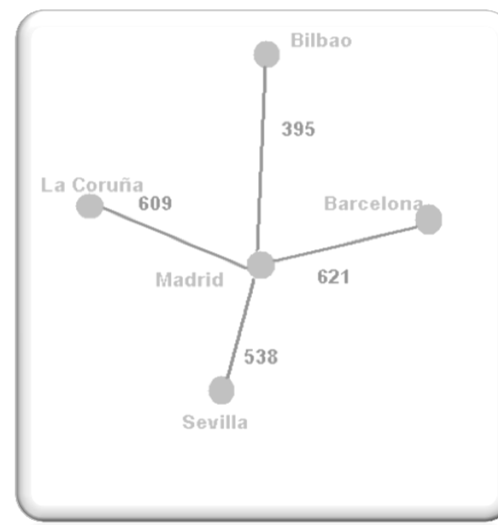
- Un grafo **G** es un par **(V,E)** donde:
  - $\mathbf{V = \{v_1, \ldots, v_n\}}$ es un conjunto de vértices
  - $\mathbf{E = \{e_1, \ldots, e_m\}}$ es un conjunto de aristas,

    con cada $\mathbf{e_k \in \{v_i, v_j\}}$, con $\mathbf{v_i, v_j \in V}$, $\mathbf{v_i \neq v_j}$
- Los vértices se representan como puntos y las aristas como líneas entre vértices
- Ejemplo:
  - $\mathbf{G = (V,E)}$
  - $\mathbf{V = \{a,b,c,d\}}$
  - $\mathbf{E = \{\{a,b\}, \{b,c\}, \{a,c\}, \{a,d\}, \{d,b\}\}}$

- Si el orden influye en la aristas se habla de **grafos dirigidos**

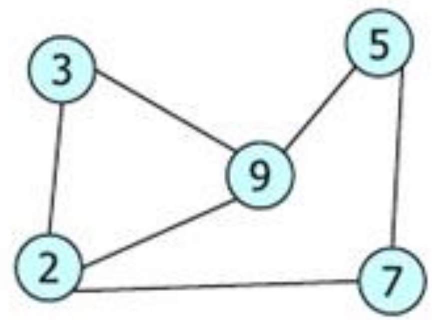


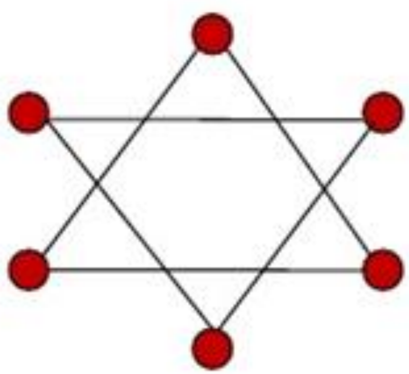
- Cuando las aristas tienen un valor numérico asociado se llama de **grafos valorados**



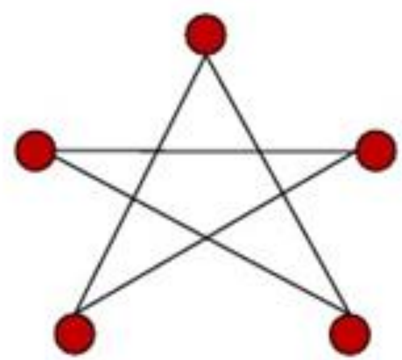
- **Grafo Conexo**
  - *Existe un **camino** entre cualquier par de nodos*

Grafo conexo





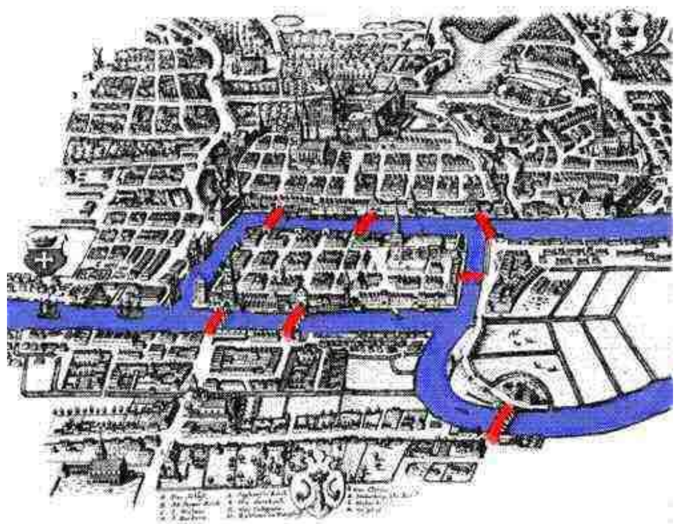
Grafo inconexo



Grafo conexo

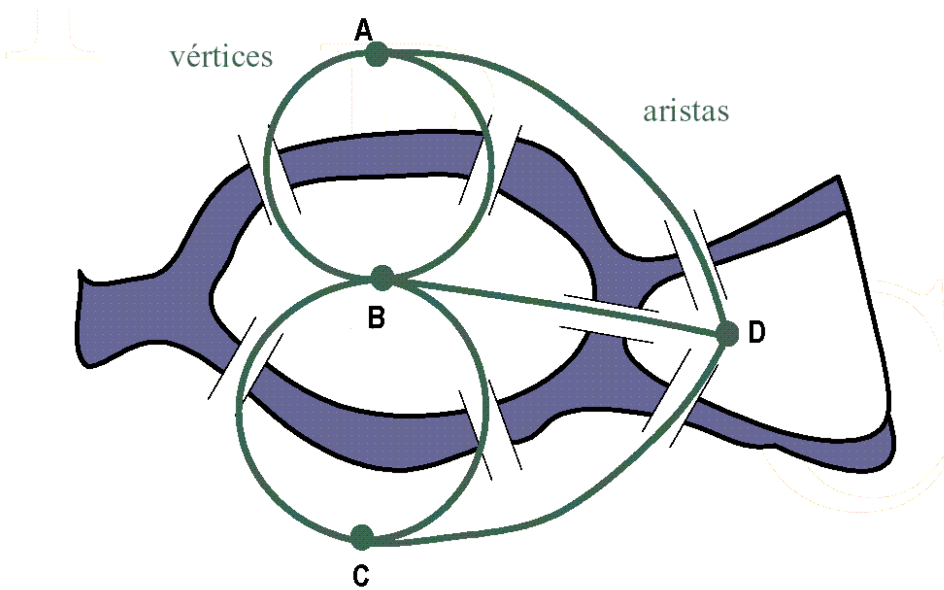
- Dos vértices se dicen **adyacentes** si existe una arista que los une
- Los vértices que forman una arista son los **extremos** de la arista
- Si **v** es un extremo de una arista **a**, se dice que **a** es **incidente** con **v**
- El grado de un vértice **v**, **gr(v)** es el número de aristas incidentes en **v.** Si hace falta indicar el grafo en el que está **v** escribiremos **gr(G,v)**

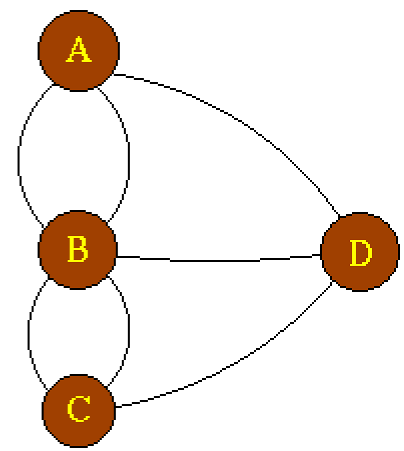
Ciudad de Könisberg, en XVIII:



¿sería posible dar un paseo pasando por cada uno de los siete puentes, sin repetir ninguno, comenzando y acabando en el mismo punto?

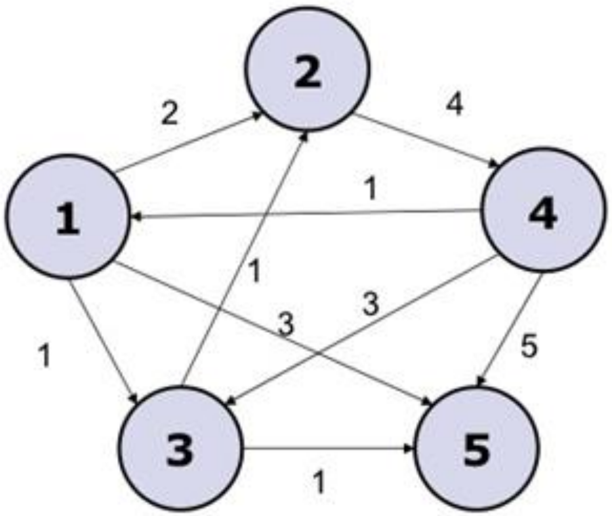
Representación propuesta por Leonard Euler en 1736:





Dijkstra

- Permite encontrar el camino más corto entre dos nodos de un grafo.



- Sea un grafo $= \{v_1; v_2; \ldots; v_n\}$ *su conjunto de vértices*
- $\Omega = (\omega_{ij})_{nxn}$ *su* matriz de pesos,
- v_p *el vértice inicial*

Dijkstra construye, en cada paso, un camino mínimo desde v_p *a otro vértice y termina cuando ha* construido uno para cada vértice (o no puede construir más)

Pseudocódigo

inicio: Ω; v_p; $L = \{v_p\}$; $D = \Omega(p, :)$

mientras sea $V - L \neq \emptyset$

    tomar $v_k \in V - L$ con $D(k)$ *mínimo*

    hacer $L = L \cup \{v_k\}$

    para cada v_j de *V-L*

        si $D(j) > D(k) + \omega_{kj}$

            hacer $D(j) = D(k) + \omega_{kj}$

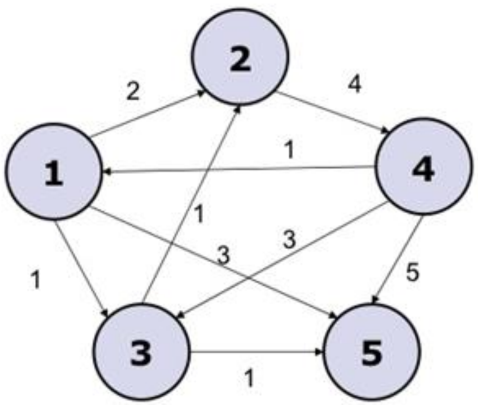
        fin

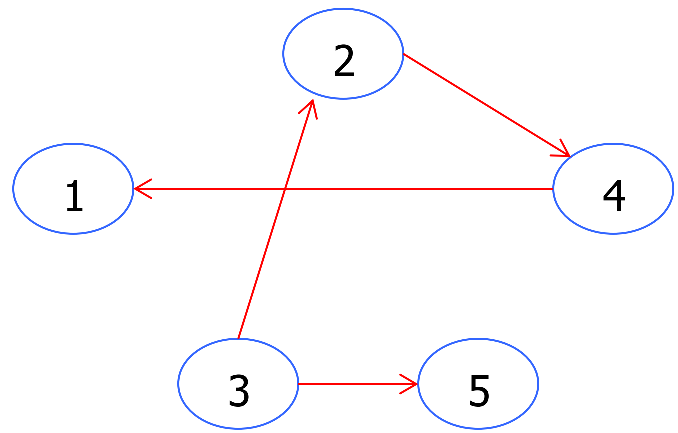
    fin

fin

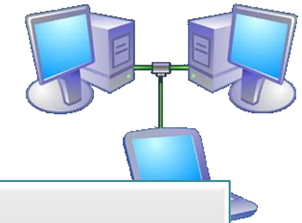
Fuente: http://www.ma.uva.es/~antonio/Industriales/Apuntes_09-10/LabM/Grafos-2010_4.pdf

Ejemplo Dijkstra para V_3





ARBOLES



Es un grafo dirigido, unidireccional, no conexo, sin ciclos, que:

Existe un nodo único –raíz- el cual no tiene arcos que provengan del árbol que entra entran en él.

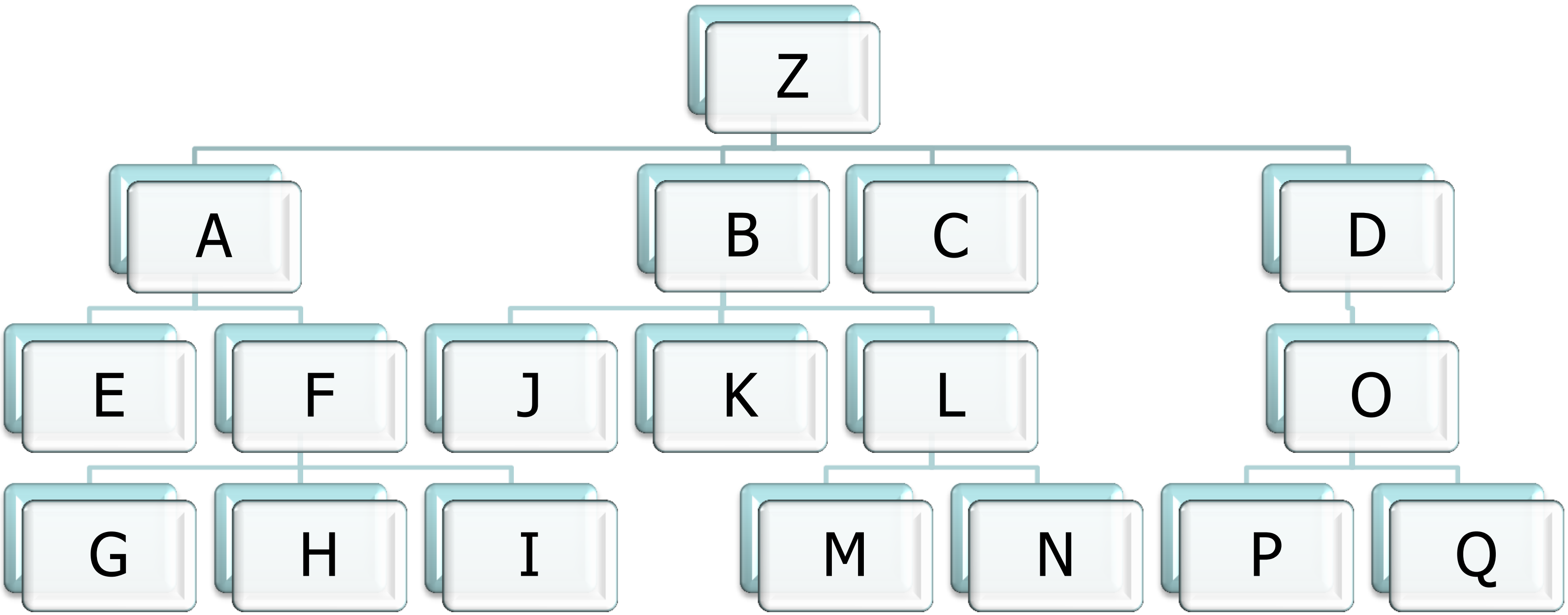
Cada uno de los nodos –excepto la raíz- tiene un arco único que entra en dicho nodo

Existe un camino único para ir a cualquier nodo del arbol.

Terminología

- Nodos Terminales (hojas). Son aquellos que no tienen sucesor.
- Sub-arboles (ramas). Es cada uno de los árboles que sale de un nodo.
- Nivel. Con un rango de 0 a n. Donde la raíz tiene nivel 0, sus sucesores tienen un nivel más, así hasta llegar a las hojas.
- Altura. Es la distancia en arcos de la raíz al nodo más lejano.
- Recorrido o Camino. Es la suma de las distancias, medidas en arcos, de la raíz a cada uno de los nodos.
- Grado: el número de hijos que tiene el elemento con más hijos dentro del árbol
- Hijo, padre, nodo interno, ….

EJEMPLO

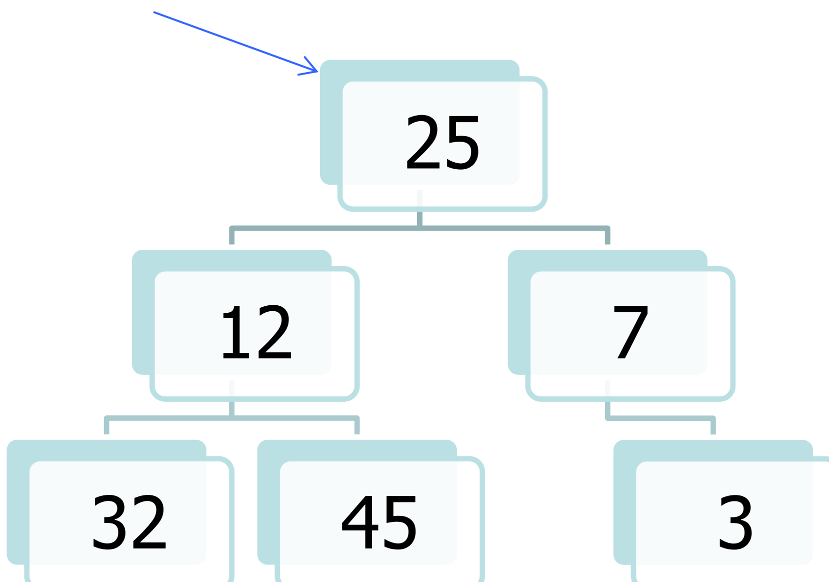


CLASIFICACIÓN

- Arboles homogéneos
- Arboles N-arios
- Arboles Completos
- Arboles Ordenados
- Arboles Balanceados
- Otros

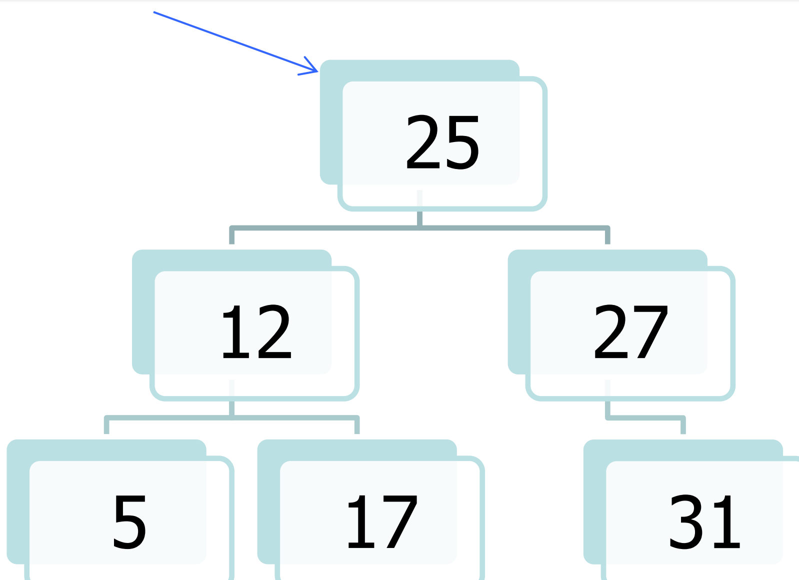
Arboles Binarios

Cada nodo esta constituido por dos enlaces, usualmente denominados izquierdo y derecho. Debe haber un criterio de ingreso



Arboles Binarios Ordenados - ABO

Un ABO, es un árbol binario ordenado, a la izquierda van los menores que el padre y a la derecha los mayores del padre.



Implementación

- Raíz igual a NULL implica árbol vacio
- Se ingresa y elimina de a un solo nodo.
- **Implementación recursiva por defecto.**
- Para trabajo con clases se tiene que trabajar en forma no recursiva, para implementación se requieren árboles y pilas.

Recorridos

- Inorden – IRD
- Post orden – IDR
- Pre Orden – RID

INORDEN - IRD	PRE ORDEN - RID	POSTORDEN - IDR

void inorden(nodo *A)
{
if(A)
{ inorden(A->izq);
cout << A->dato
<<"\t";
inorden(A->der);
}
}
	void preorden(nodo *A)
	{
	if(A)
	{ cout << A->dato
	<<"\t";
	preorden(A->izq);
	preorden(A->der);
	}
	}
		void postorden(nodo *A)
		{
		if(A)
		{ postorden(A->izq);
		postorden(A->der);
		cout << A->dato
		<<"\t";
		}
		}

```
arbol* ing_arbol(arbol *A,int x)
{  if(A==NULL)
   {
     A= new arbol;
     if(!A)
     { cout << "NO HAY SUFICIENTE MEMORIA ";
       return A;
     }
     A->dato=x;
     A->izq=A->der=NULL;
    }
   else    if(A->dato < x)
                A->der=ing_arbol(A->der,x);
            else if (A->dato > x)
                     A->izq=ing_arbol(A->izq,x);
                 else
                     {cout << "EL ELEMENTO YA EXISTE, NO PUEDE ESTAR
                               REPETIDO";   cin.get();  }
  return A;
  }
```

```
arbol* eli(arbol* A,int x)
 {   arbol *p;
   int Mayor;
   if(A->dato == x)
     {
       if(A->izq == NULL && A->der==NULL)   // caso 1: El elemento se encuentra en una hoja
                    {   delete A;
                        return NULL;
                     }
                  else if(A->izq == NULL)                // caso 2: La rama izquierda del dato es nula y en la derecha
                     { p= A->der;                         //              hay información
                       delete A;
                       return p;
                      }
                  else
                     { Mayor=MayorElem(A->izq);     // Caso 3:  Que sea un nodo interno con rama izq. y rama der.
                        A->dato = Mayor;
                        A->izq = eli(A->izq,Mayor);
                      }
      }
 else
  if (A->dato > x)                                                    // Parte recursiva para recorrer el arbol hasta encontrar elemento
                      A->izq = eli(A->izq,x);
   else
                    A->der = eli(A->der,x);
 return A;
}
```

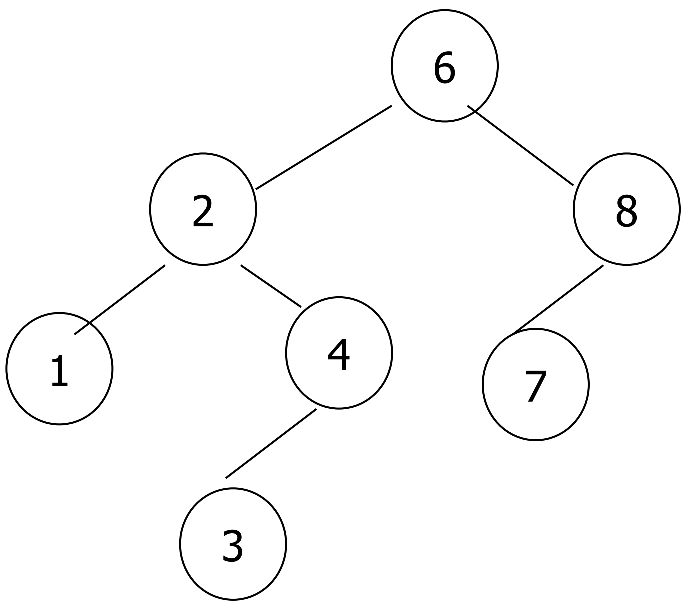
ARBOLES EQUILIBRADOS AVL

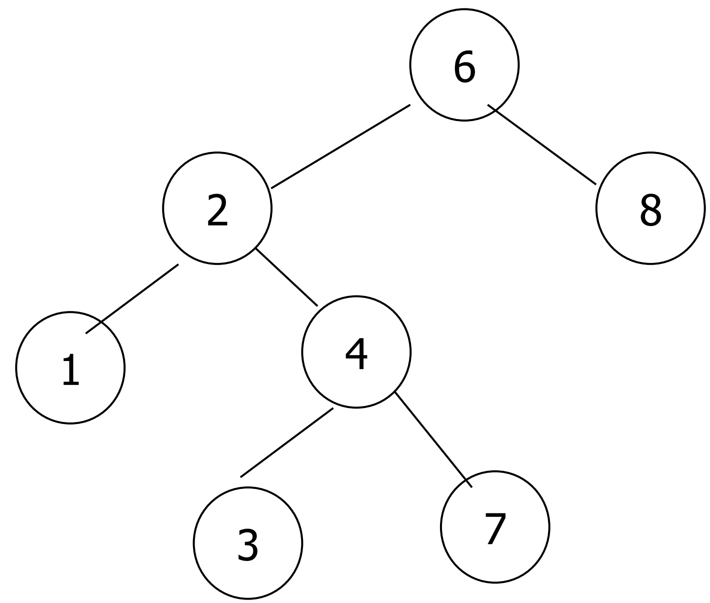
Características

- El nombre AVL son las iniciales de los hombres que idearon este tipo de árbol Adelson-Velskii y Landis en 1962.
- Arbol binario ordenado equilibrado
- O(log(n)) en el peor de los casos
- Equilibrado por altura:

$$AVL\ (A) \Leftrightarrow |h(izq) - h(der)| <= 1\ \&\&\ AVL(izq)\ \&\&\ AVL\ (der)$$

¿Cuál es AVL?





Implementación

- Agregar a estructura factor de balance (-1, 0, 1)
- Insertar como en un ABO
- Balancear – 5 casos
  - 1. Queda todo igual
  - 2. RSI
  - 3. RSD
  - 4. RDD (I)
  - 5. RDI (D)

```
struct AVL
{  int info;
   int bal;
   AVL *ri;
   AVL *rd;
};
```